Projeto Prático – 2019/2020
"Processamento de modelos gerados no *Blender*"

Já reparaste que para o desenvolvimento de jogos de computador visualmente apelativos é necessário fazer a modelação de numerosos objetos. Para isso uma das ferramentas utilizadas é o *Blender* (neste momento os alunos de TDM estão a trabalhar com o *Blender* nas aulas). Nesta ferramenta o utilizador cria os modelos que são posteriormente exportados para o motor de um jogo.
Um dos padrões para exportação destes modelos é o formato *OBJ* (ficheiros *.obj), cuja descrição detalhada podes (e deves) consultar em:

https://en.wikipedia.org/wiki/Wavefront_.obj_file

O objetivo deste projeto é calcular/determinar algumas informações acerca de objetos, a partir dos respetivos modelos previamente gravados em ficheiro no formato *OBJ*. Uma das informações fundamentais é a quantidade de material (área) de cada objeto/modelo.

Para o desenvolvimento e testes do seu projeto, são disponibilizados na página desta UC no moodle diversos ficheiros (*.obj). Assuma que cada ficheiro OBJ só tem um modelo.

Além da leitura dos modelos/ficheiros, é necessário implementar diversas funcionalidades, através de métodos cujos protótipos são apresentados abaixo. Assuma que tem de implementar as classes ***SGestao***, ***Vertice***, ***Aresta***, ***Face*** e ***Modelo***, mas que pode e deve implementar outras que considere necessárias!!! Assuma ainda que o ***SGestao*** pode ter vários modelos carregados em memória.

Funcionalidades/métodos a implementar

1. Carregar os dados de ficheiros, este método é fundamental! E pode ter várias versões!, devendo aceitar o formato *OBJ*, tanto na versão mais simples, como na versão completa (consulte atentamente a especificação do formato *OBJ* no site indicado…).

   bool SGestao::Load(const string &fich);

2. Contar o número de vértices, arestas e faces;

   int SGestao::Contar (Tipo T);

3. Determinar a área do modelo lido; o modelo já foi previamente lido. O parâmetro deste método é nome do ficheiro!, o nome do ficheiro identifica o modelo lido.

   double SGestao::AreaModelo(const string &fich);

4. Determinar envolvente a um modelo;

   bool SGestao::Envolvente(const string &fich, Ponto &Pmin, Ponto &Pmax);

5. Determinar toda a memória ocupada;

   int SGestao::Memoria();

6. Determinar a memória ocupada por um modelo.

Modelo *SGestao::Memoria(const string &fich);

7. Determinar qual o modelo que ocupa mais memória;

Modelo *SGestao::ModeloMaisMemoria();

8. Determinar o número de faces que são intersectadas por uma dada reta (a reta é definida por 2 pontos);

int SGestao::NumInterseccoes(Ponto A, Ponto B);

9. Remover um dado modelo, dado o nome do ficheiro do modelo;

bool SGestao::RemoverModelo(const string &fich);

10. Gravar para ficheiro em formato XML um dado modelo;

void SGestao::EscreverXML(const string &fich, const string &fich_xml);

11. Implemente o destrutor da classe SGEstao, que obviamente deve libertar toda a memória ocupada.

SGestao::~SGestao ();

12. Determinar a face com maior área de um Modelo;

Face *SGestao::FaceMaiorArea(const string &fich);

13. Determinar a face de um modelo onde existe maior curvatura (Será determinar a face X onde existe "maior" ângulo com as faces vizinhas de X);

Face *SGestao::FaceMaiorCurvatura(const string &fich);

Observações:

- Os alunos não devem alterar os métodos apresentados acima!.
- Podem e devem ser criados novos métodos (sempre privados ou protegidos) com os nomes que acharem por bem!
- Devem utilizador os conceitos "Orientação a objetos"

Avaliação / Observações:

- Se a gestão da memória, não estiver correta, haverá uma penalização de 5 valores;
- Se for detetado "copianço", trabalho anulado!
- Os grupos são constituídos por 1, 2 ou 3 alunos;
- Eventuais dúvidas serão esclarecidas e anexadas em documento colocado no moodle.

Entrega:

- Até ao dia 20/11//2019, os alunos devem mostrar ao prof. das práticas o método Load a funcionar;
- Época Normal – 06/01/2020;
- Época Recurso – 03/02/2020;